LOG4J - LOGGING IN DATABASE

The log4j API provides the *org.apache.log4j.jdbc.JDBCAppender* object, which is capable of putting logging information in a specified database.

JDBCAppender Configuration:

Property	Description
bufferSize	Sets the buffer size. Default size is 1.
driver	Sets the driver class to the specified string. If no driver class is specified, defaults to sun.jdbc.odbc.JdbcOdbcDriver.
layout	Sets the layout to be used. Default layout is org.apache.log4j.PatternLayout.
password	Sets the database password.
sql	Specifies SQL statement to be executed every time a logging event occurs. This could be INSERT, UPDATE, or DELETE.
URL	Sets the JDBC URL
user	Sets the database user name

Log Table Configuration:

Before you start using JDBC based logging, you shold create a table where all the log information would be maintained. Following is the SQL Statement for Creating the LOGS Table:

```
CREATE TABLE LOGS
   (USER_ID VARCHAR(20) NOT NULL,
    DATED   DATE NOT NULL,
    LOGGER  VARCHAR(50) NOT NULL,
    LEVEL   VARCHAR(10) NOT NULL,
    MESSAGE VARCHAR(1000) NOT NULL
   );
```

Sample Configuration File:

Following is a sample configuration file *log4j.properties* for JDBCAppender which will be used to log messages to a LOGS table.

```
# Define the root logger with appender file
log4j.rootLogger = DEBUG, DB

# Define the DB appender
log4j.appender.DB=org.apache.log4j.jdbc.JDBCAppender

# Set JDBC URL
log4j.appender.DB.URL=jdbc:mysql://localhost/DBNAME

# Set Database Driver
log4j.appender.DB.driver=com.mysql.jdbc.Driver
```

```
# Set database user name and password
log4j.appender.DB.user=user_name
log4j.appender.DB.password=password

# Set the SQL statement to be executed.
log4j.appender.DB.sql=INSERT INTO LOGS
                      VALUES('%x','%d','%C','%p','%m')

# Define the layout for file appender
log4j.appender.DB.layout=org.apache.log4j.PatternLayout
```

Here for a MySQL database, you would have to use actual DBNAME, user id and password where you have created LOGS table. The SQL statement is to execute an INSERT statement with the table name LOGS and values to be entered into the table.

The JDBCAppender does not need a layout to be defined explicitly. Instead, the SQL statement passed to it uses a PatternLayout

If you like to have an XML configuration file equivalent to above *log4j.properties* file, then here is the content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration>

<appender name="DB" >
   <param name="url" value="jdbc:mysql://localhost/DBNAME"/>
   <param name="driver" value="com.mysql.jdbc.Driver"/>
   <param name="user" value="user_id"/>
   <param name="password" value="password"/>
   <param name="sql" value="INSERT INTO LOGS VALUES('%x',
                            '%d','%C','%p','%m')"/>
   <layout >
   </layout>

</appender>

<logger name="log4j.rootLogger" additivity="false">
   <level value="DEBUG"/>
   <appender-ref ref="DB"/>
</logger>

</log4j:configuration>
```

Sample Program:

The following Java class is a very simple example that initializes, and then uses, the Log4J logging library for Java applications.

```
import org.apache.log4j.Logger;
import java.sql.*;
import java.io.*;
import java.util.*;

public class log4jExample{
  /* Get actual class name to be printed on */
  static Logger log = Logger.getLogger(
                      log4jExample.class.getName());

  public static void main(String[] args)
                throws IOException,SQLException{

     log.debug("Debug");
     log.info("Info");
  }
}
```

Compilation and Run:

Here are the steps to compile and run the above mentioned program. Make sure you have set PATH and CLASSPATH appropriately before proceeding for the compilation and execution.

All the libraries should be available in CLASSPATH and your *log4j.properties* file should be available in PATH. So do the following:

- Create log4j.properties as shown above.
- Create log4jExample.java as shown above and compile it.
- Execute log4jExample binary to run the program.

Now check your LOGS table inside DBNAME database and you would find following entries:

```
mysql >  select * from LOGS;
+---------+------------+--------------+-------+---------+
| USER_ID | DATED      | LOGGER       | LEVEL | MESSAGE |
+---------+------------+--------------+-------+---------+
|         | 2010-05-13 | log4jExample | DEBUG | Debug   |
|         | 2010-05-13 | log4jExample | INFO  | Info    |
+---------+------------+--------------+-------+---------+
2 rows in set (0.00 sec)
```

NOTE: Here x is used to output the NDC (nested diagnostic context) associated with the thread that generated the logging event. We use NDC to distinguish clients in server-side components handling multiple clients. Check Log4J Manual for more information on this.